REALISEZ VOTRE

ORDINATEUR INDIVIDUEL

LE but de la série d'articles que nous commençons aujourd'hui est de vous permettre l'étude théorique et surtout la réalisation d'un ordinateur individuel d'un très bon rapport qualité prix, puissant et modulaire. Ce système n'aura rien à envier aux réalisations commerciales actuelles comme vous pourrez en juger à la lecture des lignes qui suivent, tant du point de vue performances que du point de vue technologie de réalisation. De plus, l'expérience acquise par l'auteur de ces lignes dans ce genre de réalisation avec, en particulier, la série « réalisez un mini-ordinateur domestique » publiée de 1978 à 1981 dans les pages de la revue, nous permet de vous annoncer que les chances de mener à bien avec succès un tel montage seront très importantes car nous avons, fort de cette expérience, éliminé quasiment toutes les sources de problèmes qu'ont pu rencontrer certaines des personnes nous ayant suivi pour le « mini-ordinateur domestique ». Par ailleurs, et toujours en raison de l'expérience acquise, nous savons que ce genre de description rencontre un certain succès parmi les lecteurs de la revue et qu'en conséquence nous pourrons la mener à bien, ce qui n'était pas le cas il y a trois ans puisque l'expérience que nous tentions alors était la première du genre dans une revue d'électronique grand public française. Cette certitude va avoir comme conséquence une réalisation beaucoup plus logique et structurée de l'ensemble permettant très facilement son évolution ultérieure au fur et à mesure de vos désirs.

Ces précisions étant apportées et avant les quelques rappels théoriques indispensables, nous allons vous présenter quelques grandes lignes de l'ensemble.

Présentation du système

Cet ordinateur individuel est construit à partir du plus puissant microprocesseur 8 bits du marché actuel : le MC 6809 de Motorola. Ce microprocesseur a d'ailleurs été choisi par Tandy Radio Shack, pourtant habitué à la famille 8080 et Z-80, pour équiper son nouveau TRS-80 : le TRS-80 COLOR. Comme le précédent mini-ordinateur, nous allons faire appel à la technique unanimement adoptée dans l'industrie consistant à réaliser une carte par fonction ou par groupe de fonctions : cartes qui vont ensuite s'enficher dans des connecteurs en fond de panier ; connecteurs reliés par un bus banalisé. Etant donné notre décision de réaliser tous les circuits imprimés à trous métallisés, le nombre de fonctions par carte va être beaucoup plus important que dans l'ancien système au point que notre ordinateur individuel complet (avec interface pour disques souples) tiendra en trois ou quatre cartes selon l'option que vous aurez choisie. Nous allons voir ci-après la fonction de ces cartes dites de base étant entendu que nos amis lecteurs débutants peuvent laisser de côté ce passage car de nombreux termes risquent de leur être inconnus ; qu'ils se rassurent, la partie initiation suit cette présentation...

Nous avons tout d'abord la carte appelée CPU09 qui regroupe :

— Une unité centrale 6809 avec circuiterie de rafraîchissement des RAM dynamiques et gestion d'accès au bus.

— Un circuit d'interface série asynchrone dont la vitesse de transmission est programmable par logiciel de 110 à 9 600 bauds ? ce circuit est suivi des adaptateurs standard RS 232 tant sur les lignes émission/réception que sur les lignes de « handshake » RTS et CTS.

— Un triple timer 16 bits entièrement programmable, disponible totalement ou aux 2/3 (2 timers sur 3) selon le mode de fonctionnement choisi pour la carte.

— Un circuit d'interface parallèle dont une partie est configurée pour recevoir une imprimante équipée d'une interface standard type « Centronics », une autre partie étant utilisée pour l'interface cassette et la sélection du mode de fonctionnement de la carte.

— Une interface pour magnétophone à cassettes ordinaire servant à mémoriser les programmes ; interface au standard Kansass City et acceptant les cassettes utilisées sur l'ancien système.

— Une mémoire RAM d'un K-mots de 8 bits partiellement utilisée par le moniteur mais dont au moins 512 octets sont disponibles sans risque d'interférer avec les variables du moniteur.

— Une mémoire PROM de 4 K-mots de 8 bits contenant le moniteur du système et pouvant être remplacée par toute autre mémoire à votre convenance si vous souhaitez particulariser votre réalisation.

— Un prédécodage des adresses destinées aux périphériques.

— Un circuit de pagination mémoire permettant d'étendre la capacité d'adressage du 6809 de 64 K-octets à 256 K-octets.

Cette carte CPU09 constitue donc à elle seule un mini-ordinateur de base très complet puisqu'il suffit de lui raccorder un terminal quelconque pour pouvoir commencer à travailler.

Viendra ensuite une carte RAM dynamique (ne levez pas les bras au ciel, son fonctionnement est assuré sans aucun réglage grâce à une conception nouvelle des circuits). Cette carte supportera de 64 à 256 K-octets de RAM par blocs indivisibles de 64 K-octets. Le rafraîchissement des mémoires sera entièrement transparent et ne ralentira pas l'unité centrale. Les mémoires employées seront des modèles mono tension 5 V.

Nous aurons ensuite une carte d'interface disques souples

(floppy disk si vous préférez) pouvant commander de 1 à 4 « drives » simple ou double face, simple ou double densité et de format 5 pouces ou 8 pouces. A titre indicatif, rappelons qu'un disque souple 5 pouces double face, double densité peut stocker 320 K-octets et qu'un 8 pouces double face, double densité peut mémoriser 1 M-octets (1 Mega octets ou 1 million d'octets).

Vous aurez ensuite le choix, compte tenu de ce que vous souhaiterez faire, entre réaliser la carte IVG de notre ancien système (carte qui a été conçue en pensant à ce nouveau système pour lequel elle est très bien adaptée) ou réaliser une carte d'interface alphanumérique et graphique à huit couleurs avec générateur de vecteurs rapide intégré, rotateur de vecteurs, possibilité d'écriture dans tous les sens (horizontalement, verticalement, en italiques, etc.) et, de plus, haute résolution puisqu'elle fera 512 par 512 points.

Ces cartes constituent ce que nous appellerons l'ordinateur individuel standard, étant entendu que nous décrirons ensuite un certain nombre de cartes d'interfaces spécifiques très variées (convertisseurs A/D et D/A, synthétiseurs de parole, etc.).

Le logiciel sera à la mesure du matériel puisque nous vous proposerons entre autres choses : éditeur, macro-assembleur, processeur de texte, compilateur BASIC, BASIC étendu, compilateur PASCAL, jeux, etc. Mais nous verrons cela plus en détail lorsque le moment sera venu.

Technologie et composants

Afin de vous présenter des cartes supportant un nombre important de fonctions, et donc de réduire le prix de revient global du système, nous avons décidé de réaliser tous les circuits imprimés à trous métallisés. Ils seront disponibles à un prix très compétitif auprès de la société FACIM (qui réalisait déjà les cartes de l'ancien système en nous donnant toute satisfaction). Néanmoins, pour les amateurs équipés pour travailler avec la méthode photographique, nous publierons les films en vraie grandeur des deux faces de chaque circuit afin d'en permettre la reproduction, étant entendu que dans ce cas, la réalisation sera rendue plus délicate compte tenu de la non métallisation des trous, ce procédé étant irréalisable par un amateur même bien équipé.

Pour répondre à un souhait souvent formulé, nous avons fait réaliser un boîtier parfaitement adapté à cet ordinateur ; boîtier qui sera disponible entièrement monté, découpé et équipé de ses deux ventilateurs auprès de la société INCODEC. La finition en est professionnelle puisqu'il est réalisé en tôle d'acier de 1,5 mm zinguée bichromatée, hormis la face avant qui reçoit une peinture noire granitée. Les pièces mécaniques internes (supports de floppy, support de l'alimentation, bus à cartes, etc.) sont fournies prêtes à l'emploi avec ce boîtier.

Au sujet des composants, nous avons fait en sorte qu'ils soient disponibles, soit dans le commerce courant (annonceurs de la revue par exemple), soit chez FACIM pour certains circuits d'approvisionnement particulièrement délicat. A ce propos, nous voulons ouvrir une parenthèse. Le domaine de la micro-informatique évolue très vite et, bien que nous souhaitions vous faire bénéficier des derniers progrés de la technique, vous comprendrez aisément qu'il est impossible de vous présenter toujours des montages faisant appel au tout dernier produit sorti ; cela pour plusieurs raisons, tout d'abord, il faut attendre que le circuit soit distribué au niveau amateur, ce qui est parfois très long pour certains CI évolués ou spécialisés ; il faut tenir compte du temps nécessaire à l'auteur entre la sortie des caractéristiques définitives du CI et le temps de conception et de réalisation de la carte, et, surtout, il faut voir le prix du circuit qui, lors de sa sortie est toujours très élevé et chute très vite ensuite. A titre d'anecdote nous citerons un lecteur (qui n'a même pas eu le courage de signer sa lettre !) et qui, aprés avoir critiqué violemment la carte IVG de l'ancien système nous a reproché de ne pas avoir employé le nouveau circuit d'EFCIS, l'EF 9365.

A l'époque de sa lettre, ce circuit coutait 900 F. et était tout juste échantillonné ; il coute maintenant 350 F. et est disponible chez tous les revendeurs EFCIS ; c'est d'ailleurs lui que nous employons dans la carte graphique couleur précitée... sans commentaire.

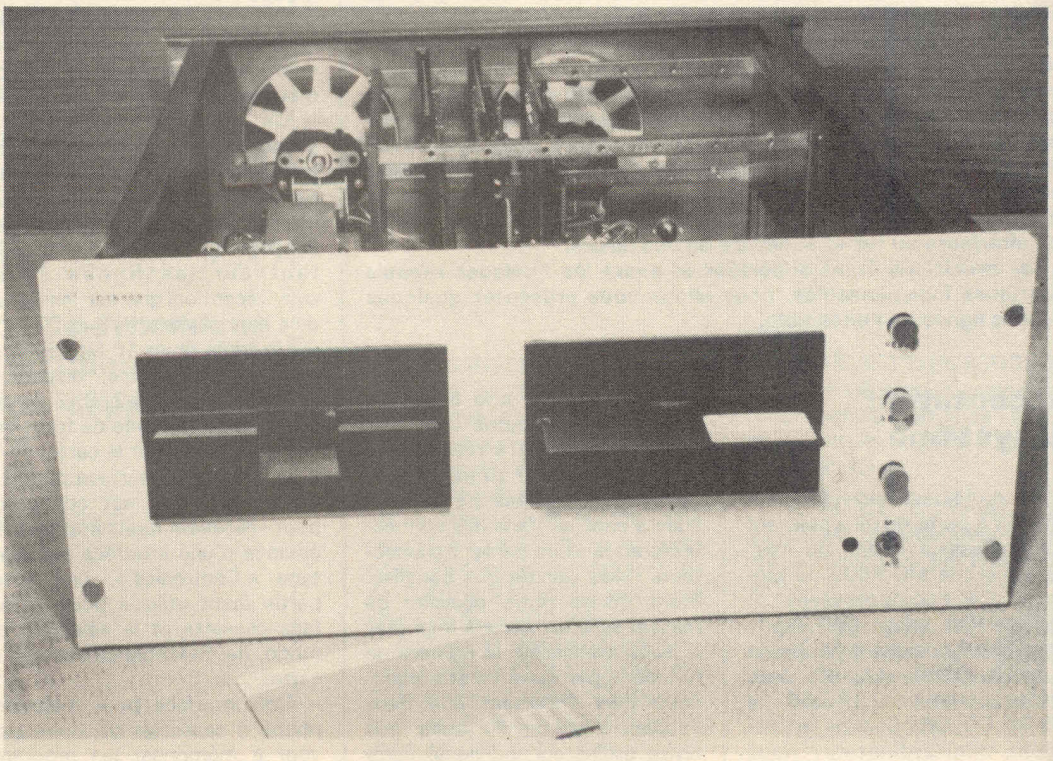
Ces indications générales étant données ; nous allons consacrer les lignes qui suivent à un peu d'initiation indispensable pour aborder cette réalisation avec une bonne vue d'ensemble.

Structure d'un mini-ordinateur

L'étude que nous allons présenter maintenant n'a pas la prétention d'être générale : en effet, nous allons l'orienter vers notre réalisation, notre propos n'étant pas de faire un cours de micro-informatique.

A ce sujet, nous rappelons à nos amis lecteurs débutants, qui souhaitent suivre cette série et qui n'ont aucune notion de micro-informatique, que, dans ce même numéro, commence à partir de ce mois, une série d'articles ayant pour but de réaliser une initiation à la micro-informatique accessible à tous les lecteurs de la revue sans connaissance préalable.

Examinons la figure 1 qui n'est autre que le synoptique simplifié de tout ordinateur quel



qu'il soit. Nous y voyons trois ensembles fondamentaux qui sont :

– L'unité centrale ou CPU en Américain qui n'est autre que la partie « pensante » de la machine ; c'est à son niveau que se font les calculs et que se prennent les décisions. Dans un mini-ordinateur, cette unité centrale est constituée d'un microprocesseur entouré de quelques boîtiers logiques.

– La mémoire qui est un ensemble de circuits capables d'emmagasiner des informations de façon temporaire, on a alors à faire à de la RAM c'est-à-dire à de la mémoire dans laquelle on peut lire et écrire ; ou de façon permanente, on a alors à faire à de la ROM c'est-à-dire de la mémoire dans laquelle on ne peut que lire. La taille de ces deux types de mémoires est très variable selon le rôle et la taille de l'ordinateur concerné, et, bien que ces deux notions soient liées, il ne faut pas systématiquement juger la puissance d'un calculateur en fonction de la taille de sa mémoire.

– Les interfaces d'entrées/ sorties qui sont les moyens de liaison entre l'unité centrale et le monde extérieur. Ces interfaces peuvent revêtir des aspects très divers, depuis le terminal clavier-écran classique (style terminal de réservation SNCF ou AIR INTER) jusqu'au convertisseur analogique digital par exemple. Le rôle de cet interface étant, dans tous les cas, de convertir des informations d'un langage en un autre.

Tous ces circuits travaillent en binaire (voir ci-après) c'est-à-dire dans un système de numération qui n'utilise que des « zéro » et des « un ». Un zéro est matérialisé par une tension voisine de la masse tandis qu'un un est matérialisé par la présence d'une tension non nulle. Comme les circuits de la famille 6800, utilisés dans notre système, travaillent avec des signaux aux normes TTL, nous utiliserons ces normes tout au long de cette réalisation (c'est par ailleurs une norme universelle) ; à savoir : un zéro logique est représenté par une tension comprise entre 0 V et 0,8 V, un un est représenté par une tension comprise entre 2,4 V et 5 V, tout signal compris entre 0,8 V et 2,4 V est à un niveau logique indéterminé.

Les différents ensembles de la figure 1 communiquent entre eux au moyen de ce que l'on appelle un bus. Le principe du bus est explicité figure 2 ; tous les signaux de même rôle (donc de même nom) de tous les boîtiers du système sont reliés entre eux, étant entendu qu'à un instant donné un seul boîtier à la fois est actif, les autres étant placés dans un état haute impédance (nous y reviendrons). Ce principe permet d'économiser une très grande quantité de fils de câblage et simplifie considérablement la réalisation.

Dans notre système (ainsi que dans beaucoup d'autres), on distingue trois bus : le bus de données, le bus d'adresses et le bus de contrôle.

Le bus de contrôle véhicule un certain nombre de signaux de service sur le rôle desquels nous aurons l'occasion de revenir plus tard ; il nous faut par contre expliciter ce que sont les adresses et les données.

Mémoire, données et adresses

Nous avons dit qu'une mémoire permettait de stocker de l'information sous forme binaire, c'est-à-dire sous forme de 0 et de 1. Cette information contenue dans une mémoire reçoit le nom très général de « donnée ». Il est bien évident que ces diverses données ne sont pas placées dans la mémoire n'importe comment puisqu'il faut que l'unité centrale, lorsqu'elle accède à la mémoire, puisse retrouver telle ou telle donnée particulière. Ces données sont donc repérées dans la mémoire par ce que l'on appelle une adresse ; un peu comme l'on repère les maisons d'une même rue au moyen d'un numéro. Dans une mémoire quelconque considérée seule, les adresses commencent toujours à zéro pour augmenter jusqu'à atteindre un nombre égal à la capacité de la mémoire. Considérons par exemple la mémoire très simple de la figure 3, c'est une mémoire de 8 mots de 2 chiffres. Nous dirons qu'à l'adresse 2 de cette mémoire se trouve la donnée 81, qu'à l'adresse 6 se trouve la donnée 27, etc.

Tout ceci pour vous faire comprendre qu'un boîtier-mémoire comporte donc obligatoirement

Adresses	Données
00	39
01	28
02	81
03	42
04	51
05	08
06	27
07	47

Fig. 3. – Exemple de mémoire ; celle-ci est une 8 mots de 2 chiffres (voir texte).

des lignes dites d'adresse servant à sélectionner au moyen du code qui leur est appliqué l'emplacement de telle ou telle donnée ; donnée que l'on peut alors lire sur les lignes dites de données de la mémoire.

Pour en revenir à nos notions de bus exposées ci-avant : notre mini-ordinateur synoptique de la figure 1 dispose donc de deux bus distincts (en plus du bus de contrôle déjà évoqué) : le bus d'adresse qui véhicule des adresses à destination des mémoires ou de circuits assimilables à des mémoires, et le bus de données qui véhicule les données se trouvant à l'adresse transmise par le bus d'adresses. Le sens de transfert des signaux d'adresse est toujours le même et va du microprocesseur vers la mémoire et les

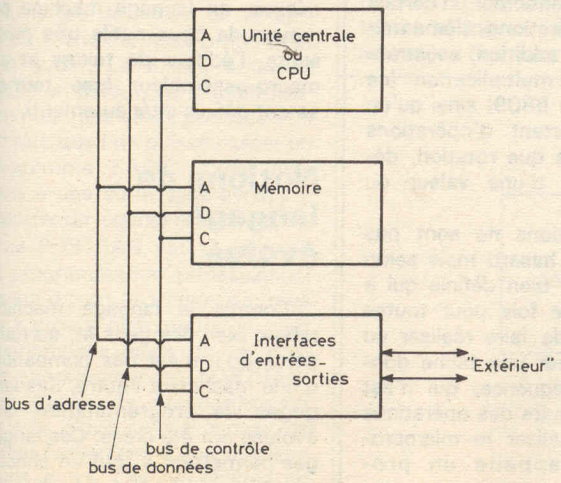


Fig. 1. – Synoptique très général de tout ordinateur.

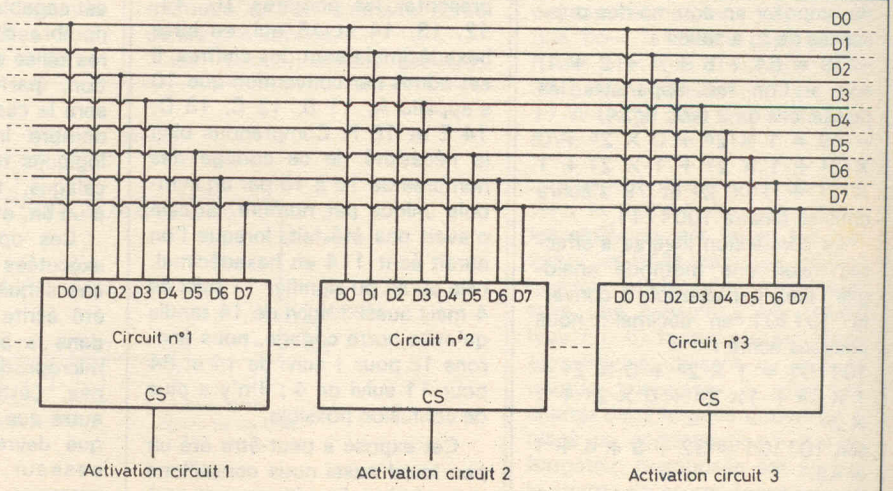


Fig. 2. – Principe général d'une connexion par bus.

périphériques puisque le seul élément doué de décision est le microprocesseur ; par contre, les données peuvent se déplacer dans les deux sens puisque le microprocesseur peut lire des données dans une mémoire (auquel cas les données vont dans le sens mémoire vers micro) mais il peut aussi écrire dans une mémoire (auquel cas les données vont dans le sens micro vers mémoire).

Ceci étant vu, nous allons procéder à quelques rappels de numération car, si le binaire est bien connu de nombreux lecteurs, il n'en est pas de même pour l'hexadécimal.

Rappels de numération

La numération binaire est très simple dans sa théorie comme dans son principe puisqu'il suffit de travailler en base 2 plutôt qu'en base 10. Qu'est ce que cela signifie ? Tout simplement qu'en numération décimale on exprime les nombres comme des sommes de multiples de puissance de 10 ; ainsi, par exemple, quand nous écrivons 389, cela signifie :

– $389 = 3 \times 100 + 8 \times 10 + 9 \times 1$, ou encore, si l'on fait intervenir les puissances de 10 (voir tableau de la fig. 4).

– $389 = 3 \times 10^2 + 8 \times 10^1 + 9 \times 10^0$.

En numération binaire, on exprime les nombres à partir des puissances de 2. Soit, par exemple, à convertir en binaire 79, nous allons commencer par le décomposer en somme des puissances de 2, à savoir :

– $79 = 64 + 8 + 4 + 2 + 1$, soit si l'on fait apparaître les puissances de 2 (voir fig. 4)

– $79 = 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$ et 79 s'écrira donc en binaire 1001111.

La conversion inverse s'effectue selon une méthode analogue ; soit, par exemple à convertir 101101 en décimal ; nous pouvons écrire :

$101101 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

soit $101101 = 32 + 8 + 4 + 1 = 45$.

Ce procédé de conversion, quoique simple, présente cependant un défaut il nécessite des calculs d'autant plus longs que les nombres à convertir sont grands. Par ailleurs, on peut remarquer que pour coder en binaire les chiffres de 0 à 9 il faut 4 chiffres binaires (on dit 4 bits, de l'Américain BInary digiT) allant de 0000 à 1001, or, l'on constate également que 4 bits permettent de réaliser 16 combinaisons différentes : donc nos quatre bits sont sous employés pour coder 10 chiffres seulement puisqu'ils peuvent coder de 0 à 15 inclus.

n	2^n	10^n	16^n
0	1	1	1
1	2	10	16
2	4	100	256
3	8	1 000	4 096
4	16	10 000	65 536
5	32	100 000	1 048 576
6	64	1 000 000	–
7	128	10 000 000	–
8	256	100 000 000	–
9	512	1 000 000 000	–
10	1 024	–	–
11	2 048	–	–
12	4 096	–	–
13	8 192	–	–

Fig. 4. – Tableau des puissances de 2, 10 et 16.

Tout cela nous conduit à introduire une nouvelle base de numération qui n'est autre que la base 16 appelée aussi base hexadécimale. Cette base s'utilise comme toutes les autres et l'on passe du décimal à l'hexadécimal et vice versa comme expliqué ci-avant pour le binaire, les puissances de 10 ou de 2 étant ici des puissances de 16. Par ailleurs, comme il n'existe pas de symbole pour représenter les nombres 10, 11, 12, 13, 14 et 15 qui, en base hexadécimale sont des chiffres, il est admis par convention que 10 s'appelle A, 11 B, 12 C, 13 D, 14 E et 15 F. Comprenons bien la nécessité de ce codage des nombres de 10 à 15 par un symbole unique par nombre. Si cela n'avait pas été fait, lorsque l'on aurait écrit 114 en hexadécimal, cela aurait pu signifier 11 suivi de 4 mais aussi 1 suivi de 14 tandis qu'avec notre codage, nous écrirons 1E pour 1 suivi de 14 et B4 pour 11 suivi de 4 ; il n'y a plus de confusion possible.

Cet exposé a peut-être été un peu lourd aussi nous conseillons vous de le relire doucement et à tête reposée afin de bien assimiler le principe de la numération binaire. Vous pouvez d'ailleurs vous exercer à convertir quelques nombres décimaux en hexadécimal et vice versa.

Rassurez-vous, il ne vous sera pas nécessaire de connaître l'hexadécimal pour utiliser votre ordinateur individuel, mais, comme nous serons amenés à en parler lors de l'étude théorique, nous préférions faire ces quelques explications dès le début.

Microprocesseur et notions de programme

Cet assemblage de circuits reliés entre eux par des bus et échangeant de l'information binaire est très élégant, encore faut-il savoir comment cela peut arriver à fonctionner. Pour cela, sachez que le cœur du système, c'est-à-dire le microprocesseur, est capable d'effectuer un certain nombre d'opérations élémentaires telles que addition, soustraction, parfois multiplication (ce sera le cas du 6809) ainsi qu'un nombre important d'opérations logiques telles que rotation, décalages, test d'une valeur ou d'un bit, etc.

Ces opérations ne sont pas exécutées au hasard mais selon une séquence bien définie qui a été écrite une fois pour toutes dans le but de faire réaliser au microprocesseur une tâche donnée. Cette séquence, qui n'est autre que la suite des opérations que devra réaliser le microprocesseur, s'appelle un programme. Vous comprenez donc dès cet instant que la soi-disant « intelligence de l'ordinateur », comme disent certains, n'est en fait que l'intelligence du programmeur ayant écrit la suite d'instructions à exécuter. Le seul avantage de l'ordinateur étant qu'il travaille très vite et qu'il peut donc accomplir des tâches répétitives ou utilisant un très grand nombre de paramètres en un temps très bref.

Ce programme, comme tout le reste, est inscrit en mémoire sous forme... binaire (vous l'aviez deviné), chaque opération que peut et que sait exécuter le microprocesseur ayant un code particulier appelé (comme c'est original) le code opération ou le code machine.

Ces codes machines sont propres à chaque type de microprocesseur et sont totalement incompatibles d'une machine à l'autre. Cela signifie qu'un programme écrit pour un 8080 ne tournera pas sur un 6800 et vice versa. Cet inconvénient a été compensé par l'introduction de ce que l'on appelle les langages évolués (voir ci-après). Avant de voir ceux-ci sachez cependant que ce langage machine, bien qu'assez rébarbatif au premier abord est le plus puissant langage de programmation qui soit ; en effet, étant donné qu'il est adapté au microprocesseur puisque c'est en fait une partie intégrante de celui-ci, les programmes y faisant appel seront les plus courts et les plus rapides qu'il sera possible de réaliser pour une application donnée. Sachez aussi que notre ordinateur individuel, contrairement à certaines machines du commerce vous permettra de travailler, si vous le désirez, en langage machine au moyen de deux outils très puissants, l'éditeur de textes et un macro-assembleur (ces termes seront définis ultérieurement).

Notions de langage évolué

Comme le langage machine rebute bien des gens et, surtout, parce qu'il n'est pas compatible d'une machine à l'autre, des langages de programmation dits évolués ont été créés. Ces langages permettent à tout un chacun ayant appris leur vocabulaire de base (très simple pour certains

langages tel le BASIC) d'écrire des programmes qui pourront fonctionner sur tous les ordinateurs existants, sous réserve qu'ils acceptent le langage choisi. En réalité, si l'on va au fond des choses, un ordinateur auquel vous faites exécuter un programme en langage évolué traduit automatiquement, et sans même que vous vous en aperceviez, ce programme en langage machine puis exécute ensuite ce qu'il a ainsi traduit étant donné que, de toute façon, il ne sait exécuter que son propre code machine. Bien que cette phase de traduction ne vous apparaisse pas ; elle existe bel et bien et se traduit par une perte de temps qui, si elle peut être négligée pour les petits programmes, peut atteindre de très grandes proportions pour les programmes longs et complexes. A titre d'anecdote, nous avons un jeu d'échecs qui réfléchit en moyenne 5 minutes par coup lorsqu'il est écrit en BASIC (un des langages évolués) alors qu'il lui faut seulement 30 secondes si on utilise la version écrite en langage machine.

Les noms des langages évolués classiques doivent vous être familiers :

– Le BASIC est le langage le plus répandu chez les amateurs ; c'est un langage qui a été conçu, à l'origine, pour des gens qui n'avaient aucune notion de programmation ; il est donc très facile à aborder et à employer. Très longtemps ignoré pour des applications professionnelles, il commence à refaire surface en raison des nombreuses possibilités dont sont pourvus les BASIC actuels (au détriment de la simplicité d'emploi, mais on ne peut tout avoir). Le BASIC à quand même un défaut, sa standardisation n'est pas assez poussée et il est rare que l'on puisse passer un programme BASIC d'une machine à une autre sans avoir à y apporter de légères retouches.

– Le FORTRAN est le langage de programmation professionnel standard ; on le trouve sur tous les « gros » ordinateurs et tout programmeur digne de ce nom sait programmer en FORTRAN. Son niveau de standardisation est très élevé ce qui permet de passer facilement d'une machine à une autre. Les possibilités en sont étendues mais son utilisation est nettement moins simple que le BASIC et il faut une longue période d'adaptation pour l'utiliser de façon efficace. Ce langage ne se rencontre qu'exceptionnelllement sur du matériel amateur.

– Le PASCAL est un langage relativement nouveau (cinq ans environ) pour simplifier, nous pouvons dire qu'il présente les avantages du BASIC et du FORTRAN sans en avoir les inconvénients. Malheureusement, sa standardisation est encore insuffisante puisque seul un « noyau » de PASCAL est standard, chacun s'ingéniant à ajouter sa touche personnelle.

– Le COBOL, l'ALGOL, le PL1, etc., sont des langages très spécifiques et à usage trop particularisé pour que nous nous y arrêtions ; ils ne se rencontrent, par ailleurs, jamais sur des machines d'amateurs.

Et notre ordinateur individuel dans tout ça, que pourra-t-il faire ? Eh bien, il travaillera, si vous le désirez, en BASIC très évolué (comparable au BASIC étendu sur disquette de notre ancien système) ou en PASCAL ; ceci étant vrai au moment où nous écrivons ces lignes et pouvant évoluer (en mieux) dans l'avenir en fonction de votre intérêt pour cette réalisation. Nous reviendrons, bien évidemment sur ces langages le moment opportun et nous allons passer maintenant à autre chose.

Les différents types de mémoires

Ainsi que nous l'avons évoqué ci-avant, il existe deux grands types de mémoires, celles dans lesquelles l'on peut lire et écrire des données que l'on appelle des RAM et celles où l'on ne peut que lire des données et que l'on appelle des ROM.

Les RAM (de l'Américain Random Acces Memory ce qui signifie mémoire à accès aléatoire) sont donc des mémoires dans lesquelles le microprocesseur va pouvoir lire et écrire des données à tout instant et autant de fois que cela sera nécessaire. Ces mémoires constituent en général les mémoires de travail et l'on y stocke des variables à caractère temporaire. En effet, à la coupure de leur alimentation, les RAM perdent leur contenu ; de plus, à la mise sous tension, une RAM contient des données aléatoires (encore que cela dépende du type exact de mémoire). Les RAM sont divisées en deux grandes familles : les RAM statiques et les RAM dynamiques. Nous ferons appel aux deux familles car, comme nous allons le voir, elles présentent toutes deux des avantages et des inconvénients. Les RAM statiques sont très simples d'emploi : la figure 5 montre à cet effet un boîtier standard et l'on y voit :

– Des lignes d'adresses allant de A0 à A10 dans cet exemple.

– Des lignes de données allant de D0 à D7 dans cet exemple.

– Une ligne d'activation du boîtier appelée CS (pour Chip Select ce qui signifie sélection de la « puce ») qui permet d'activer ou non le boîtier facilitant sa connexion sur une structure de type bus évoquée ci-avant.

– Une ligne R/W (pour Read/Write ce qui signifie lecture écriture) qui sert à indiquer au boîtier mémoire si l'on souhaite écrire ou lire une donnée.

Le fonctionnement est très simple : l'on sélectionne le boîtier en mettant CS au niveau adéquat (bas en général), on applique l'adresse à laquelle on souhaite agir sur A0 à A10, on positionne R/W selon ce que l'on veut faire et, soit on lit la donnée contenue à l'adresse demandée sur D0 à D7, soit on place sur D0 à D7 la donnée à écrire à l'adresse spécifiée. Cette succession d'événements se faisant selon un chronogramme bien précis que nous n'évoquerons point ici. Tant que l'on ne coupe pas l'alimentation, le boîtier RAM statique conserve l'information qui y a été placée. Avec la RAM dynamique, le principe de fonctionnement est analogue, mais, pour que la mémoire conserve l'information qui y a été placée, il faut périodiquement effectuer une opération appelée rafraîchissement de la RAM dynamique, opération qui porte bien son nom si on la compare à l'expression populaire « rafraîchir la mémoire » ; en effet, ce rafraîchissement a pour but de maintenir en bon état l'information qui a été emmagasinée dans la mémoire. Il est évident, même si nous n'entrons pas plus dans le vif du sujet que la nécessité de ce rafraîchissement périodique complique de façon importante la mise en œuvre de ces mémoires. Alors pourquoi en utiliser ? Tout simplement parce que, à prix égal et surtout à encombrement égal, les RAM dynamiques ont une capacité quatre fois supérieure à celle des RAM statiques et une consommation nettement inférieure. Ainsi, notre ordinateur équipé de 64 K-octets de RAM dynamique va utiliser 8 boîtiers consommant en tout 1,6 W alors que pour la même capacité en RAM statique il aurait fallu 32 boîtiers consommant en tout 17 W (soit plus de 3 A sous 5 V).

Ici encore, il n'y a pas de mystère, le rapport de quatre entre les capacités est lié au fait suivant : les RAM statiques utilisent comme cellule mémoire élémentaire une bascule à transistors (voir fig. 6) tandis que les RAM dynamiques utilisent un seul et unique transistor à effet de champ dans la grille duquel on a placé un condensateur ; et l'information mémorisée est en fait la charge de ce condensateur, ce qui explique qu'il faille régulièrement régénérer sa charge afin de

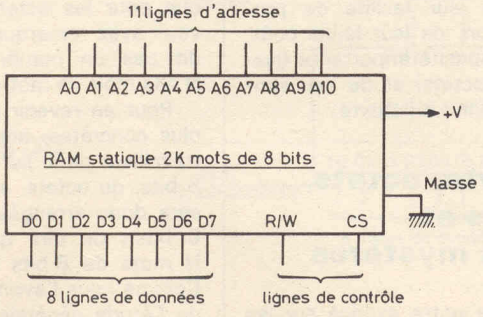


Fig. 5. – Les signaux dont on dispose sur un boîtier de RAM statique de 2 K-mots de 8 bits.

langages tel le BASIC) d'écrire des programmes qui pourront fonctionner sur tous les ordinateurs existants, sous réserve qu'ils acceptent le langage choisi. En réalité, si l'on va au fond des choses, un ordinateur auquel vous faites exécuter un programme en langage évolué traduit automatiquement, et sans même que vous vous en aperceviez, ce programme en langage machine puis exécute ensuite ce qu'il a ainsi traduit étant donné que, de toute façon, il ne sait exécuter que son propre code machine. Bien que cette phase de traduction ne vous apparaisse pas ; elle existe bel et bien et se traduit par une perte de temps qui, si elle peut être négligée pour les petits programmes, peut atteindre de très grandes proportions pour les programmes longs et complexes. A titre d'anecdote, nous avons un jeu d'échecs qui réfléchit en moyenne 5 minutes par coup lorsqu'il est écrit en BASIC (un des langages évolués) alors qu'il lui faut seulement 30 secondes si on utilise la version écrite en langage machine.

Les noms des langages évolués classiques doivent vous être familiers :

– Le BASIC est le langage le plus répandu chez les amateurs ; c'est un langage qui a été conçu, à l'origine, pour des gens qui n'avaient aucune notion de programmation ; il est donc très facile à aborder et à employer. Très longtemps ignoré pour des applications professionnelles, il commence à refaire surface en raison des nombreuses possibilités dont sont pourvus les BASIC actuels (au détriment de la simplicité d'emploi, mais on ne peut tout avoir). Le BASIC à quand même un défaut, sa standardisation n'est pas assez poussée et il est rare que l'on puisse passer un programme BASIC d'une machine à une autre sans avoir à y apporter de légères retouches.

– Le FORTRAN est le langage de programmation professionnel standard ; on le trouve sur tous les « gros » ordinateurs et tout programmeur digne de ce nom sait programmer en FORTRAN. Son niveau de standardisation est très élevé ce qui permet de passer facilement d'une machine à une autre. Les possibilités en sont étendues mais son utilisation est nettement moins simple que le BASIC et il faut une longue période d'adaptation pour l'utiliser de façon efficace. Ce langage ne se rencontre qu'exceptionnelllement sur du matériel amateur.

– Le PASCAL est un langage relativement nouveau (cinq ans environ) pour simplifier, nous pouvons dire qu'il présente les avantages du BASIC et du FORTRAN sans en avoir les inconvénients. Malheureusement, sa standardisation est encore insuffisante puisque seul un « noyau » de PASCAL est standard, chacun s'ingéniant à ajouter sa touche personnelle.

– Le COBOL, l'ALGOL, le PL1, etc., sont des langages très spécifiques et à usage trop particularisé pour que nous nous y arrêtions ; ils ne se rencontrent, par ailleurs, jamais sur des machines d'amateurs.

Et notre ordinateur individuel dans tout ça, que pourra-t-il faire ? Eh bien, il travaillera, si vous le désirez, en BASIC très évolué (comparable au BASIC étendu sur disquette de notre ancien système) ou en PASCAL ; ceci étant vrai au moment où nous écrivons ces lignes et pouvant évoluer (en mieux) dans l'avenir en fonction de votre intérêt pour cette réalisation. Nous reviendrons, bien évidemment sur ces langages le moment opportun et nous allons passer maintenant à autre chose.

Les différents types de mémoires

Ainsi que nous l'avons évoqué ci-avant, il existe deux grands types de mémoires, celles dans lesquelles l'on peut lire et écrire des données que l'on appelle des RAM et celles où l'on ne peut que lire des données et que l'on appelle des ROM.

Les RAM (de l'Américain Random Acces Memory ce qui signifie mémoire à accès aléatoire) sont donc des mémoires dans lesquelles le microprocesseur va pouvoir lire et écrire des données à tout instant et autant de fois que cela sera nécessaire. Ces mémoires constituent en général les mémoires de travail et l'on y stocke des variables à caractère temporaire. En effet, à la coupure de leur alimentation, les RAM perdent leur contenu ; de plus, à la mise sous tension, une RAM contient des données aléatoires (encore que cela dépende du type exact de mémoire). Les RAM sont divisées en deux grandes familles : les RAM statiques et les RAM dynamiques. Nous ferons appel aux deux familles car, comme nous allons le voir, elles présentent toutes deux des avantages et des inconvénients. Les RAM statiques sont très simples d'emploi : la figure 5 montre à cet effet un boîtier standard et l'on y voit :

– Des lignes d'adresses allant de A0 à A10 dans cet exemple.

– Des lignes de données allant de D0 à D7 dans cet exemple.

– Une ligne d'activation du boîtier appelée CS (pour Chip Select ce qui signifie sélection de la « puce ») qui permet d'activer ou non le boîtier facilitant sa connexion sur une structure de type bus évoquée ci-avant.

– Une ligne R/W (pour Read/Write ce qui signifie lecture écriture) qui sert à indiquer au boîtier mémoire si l'on souhaite écrire ou lire une donnée.

Le fonctionnement est très simple : l'on sélectionne le boîtier en mettant CS au niveau adéquat (bas en général), on applique l'adresse à laquelle on souhaite agir sur A0 à A10, on positionne R/W selon ce que l'on veut faire et, soit on lit la donnée contenue à l'adresse demandée sur D0 à D7, soit on place sur D0 à D7 la donnée à écrire à l'adresse spécifiée. Cette succession d'événements se faisant selon un chronogramme bien précis que nous n'évoquerons point ici. Tant que l'on ne coupe pas l'alimentation, le boîtier RAM statique conserve l'information qui y a été placée. Avec la RAM dynamique, le principe de fonctionnement est analogue, mais, pour que la mémoire conserve l'information qui y a été placée, il faut périodiquement effectuer une opération appelée rafraîchissement de la RAM dynamique, opération qui porte bien son nom si on la compare à l'expression populaire « rafraîchir la mémoire » ; en effet, ce rafraîchissement a pour but de maintenir en bon état l'information qui a été emmagasinée dans la mémoire. Il est évident, même si nous n'entrons pas plus dans le vif du sujet que la nécessité de ce rafraîchissement périodique complique de façon importante la mise en œuvre de ces mémoires. Alors pourquoi en utiliser ? Tout simplement parce que, à prix égal et surtout à encombrement égal, les RAM dynamiques ont une capacité quatre fois supérieure à celle des RAM statiques et une consommation nettement inférieure. Ainsi, notre ordinateur équipé de 64 K-octets de RAM dynamique va utiliser 8 boîtiers consommant en tout 1,6 W alors que pour la même capacité en RAM statique il aurait fallu 32 boîtiers consommant en tout 17 W (soit plus de 3 A sous 5 V).

Ici encore, il n'y a pas de mystère, le rapport de quatre entre les capacités est lié au fait suivant : les RAM statiques utilisent comme cellule mémoire élémentaire une bascule à transistors (voir fig. 6) tandis que les RAM dynamiques utilisent un seul et unique transistor à effet de champ dans la grille duquel on a placé un condensateur ; et l'information mémorisée est en fait la charge de ce condensateur, ce qui explique qu'il faille régulièrement régénérer sa charge afin de

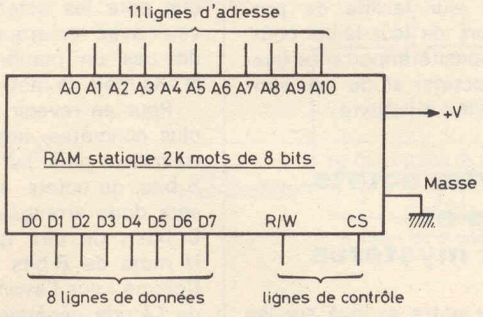


Fig. 5. – Les signaux dont on dispose sur un boîtier de RAM statique de 2 K-mots de 8 bits.

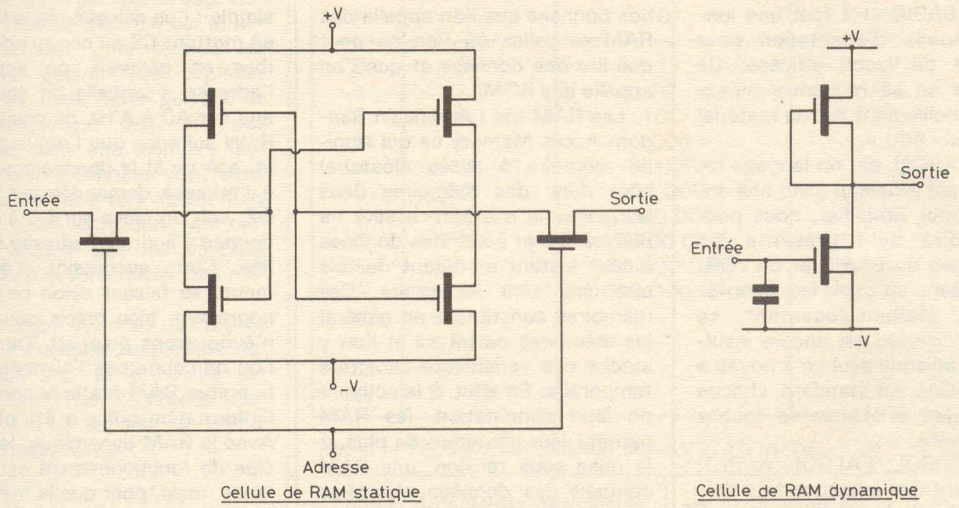


Fig. 6. – Schéma d'une cellule mémoire élémentaire dans le cas d'une RAM statique et d'une RAM dynamique.

ne pas perdre cette information. Après ce tour d'horizon rapide des RAM (nous y reviendrons en temps utile) ; passons aux ROM.

Les ROM, de l'Américain Read Only Memory (ou mémoire à lecture seulement) sont donc des mémoires qui ne peuvent qu'être lues par le microprocesseur, cela signifie qu'il a fallu, à un moment ou à un autre, aller y écrire pour y placer, de manière définitive, de l'information. C'est cette façon d'y placer l'information initiale qui différencie les ROM entre elles. On distingue en effet deux grandes familles de ROM : les ROM programmables par masque et les PROM. Avant de poursuivre, précisons que l'opération consistant à écrire l'information initiale à laquelle nous venons de faire allusion dans une ROM s'appelle la programmation de celle-ci. Les ROM programmables par masque s'éloignent un peu de notre propos, en effet, ces mémoires ne peuvent qu'être programmées par leur fabricant lors de la réalisation du circuit intégré. Il est donc bien évident que ce procédé est réservé aux utilisateurs qui achètent plusieurs centaines de ROM contenant « la même chose ». Ces ROM ont cependant un avantage certain ; elles ont une capacité très importante puisque l'on sait faire jusqu'à 8 K-octets.

Les PROM nous intéressent plus, en effet, ce sont des ROM programmables (sous certaines conditions) par l'utilisateur. Elles sont scindées en deux groupes : les PROM à fusibles et les REPROM ou UVPROM (les EAROM rentrent aussi dans cette famille mais nous aurons l'occasion d'en parler plus avant dans cette réalisation). Les PROM à fusibles sont des mémoires que l'on peut, au moyen d'un programmeur adéquat, programmer une fois pour toutes ; la programmation consistant à faire « sauter » des micro fusibles contenus dans le circuit intégré. Les REPROM par contre peuvent aussi être programmées par un programmeur adéquat mais elles disposent en plus d'une possibilité d'effacement en exposant la puce de la mémoire (placée à cet effet sous une fenêtre en quartz) à un rayonnement ultraviolet de longueur d'onde bien définie. Ces mémoires REPROM ont un grand succès auprès des amateurs et nous y ferons largement appel en raison de leur facilité de programmation, de leur faible coût, de leur capacité importante (jusqu'à 4 K-octets) et de leur simplicité de mise en œuvre.

Bit, byte, octets, K.mots et autres mystères

Lors de notre exposé sur les mémoires, vous avez vu apparaître plusieurs fois le mot K-octets lorsque nous parlions de la capacité des mémoires. Le moment est venu de vous donner quelques explications à ce sujet ainsi que sur les mots qui ornent le titre de ce paragraphe. Nous avons déjà dit, lors des rappels de numération, qu'un chiffre binaire s'appelait un bit (de l'Anglais BInary digiT). Dans notre ordinateur individuel, comme dans la très grande majorité des « petites machines », les données sur lesquelles travaille le microprocesseur sont des mots de 8 bits (encore que, dans notre cas, le 6809 sache manipuler des mots de 16 bits, nous y reviendrons). Un mot de 8 bits s'appelle un octet dans la littérature française et un byte dans la littérature d'outre Atlantique (attention bit se prononce bit, byte se prononce « baillte »). Lorsqu'un mot binaire comporte un nombre de bits autre que 8, on dit que c'est un mot de N bits : il n'y a d'appellation particulière que pour les octets. A propos, vous avez remarqué que l'on ne dit pas un nombre binaire de N bits mais un mot de N bits.

Pour en revenir à des choses plus concrètes, notre ordinateur travaillera donc sur des mots de 8 bits, ou octets, et sa mémoire sera donc arrangée en mots de 8 bits ; on dira qu'elle « fait » N mots de 8 bits ou N octets. Comme nous l'avons exposé lors de l'étude générale de la structure d'un ordinateur, en début d'article, pour retrouver une donnée dans une mémoire, il faut connaître son adresse ; cette adresse est, bien sûr, en binaire et, dans la plupart des ordinateurs à usage personnel est codée sur 16 bits, cela signifie qu'elle pourra varier de 0000 à FFFF (en hexadécimal) ou de 0000 à 65535 si l'on parle en décimal (faites la conversion, c'est un bon exercice !). Dans notre mini-ordinateur, nous disposerons de 18 lignes d'adresse ce qui signifie que nous pourrons aller jusqu'à 262 143 ou 3FFFF si vous préférez. Pour simplifier un peu les choses, et pour parler facilement de taille mémoire, on introduit dans le langage de la micro-informatique une « unité » qui est le K ou K-mots ou K-octets. Par analogie avec mètre et km vous pourriez penser que un K-octets c'est mille octets ; eh bien, c'est presque cela. En effet, 1 000 ne se code pas naturellement de manière simple en binaire ; par contre 1 024 se code très bien puisque c'est 100 000 000 en binaire ou 400 en hexadécimal. L'« unité » de mesure de la mémoire est donc le K-octets qui fait 1 024 octets et l'on parle ainsi de mémoires de 8 K-mots de 8 bits, de 4 K-mots de 8 bits, etc., ce qui veut dire respectivement de 8 192 mots de 8 bits et de 4 096 mots de 8 bits. Les mini-ordinateurs traditionnels ont donc une capacité d'adressage maximale de 64 K-octets (encore que l'on puisse très rarement exploiter à fond cette possibilité) et notre ordinateur individuel aura une capacité mémoire maximale de 256 K-octets. C'est plus que suffisant, même pour une application semi-professionnelle.

Circuits d'interface parallèle et série

Ainsi que nous l'avons dit, les circuits d'interfaces permettent à l'unité centrale d'un ordinateur de dialoguer avec le monde extérieur ou, tout simplement avec l'utilisateur du système. Il existe d'innombrables circuits d'interface selon la source de données qui doit dialoguer avec l'ordinateur. Par exemple, dans un voltmètre numérique programmable, les interfaces d'entrée sont des convertisseurs analogiques digitaux qui transforment en binaire

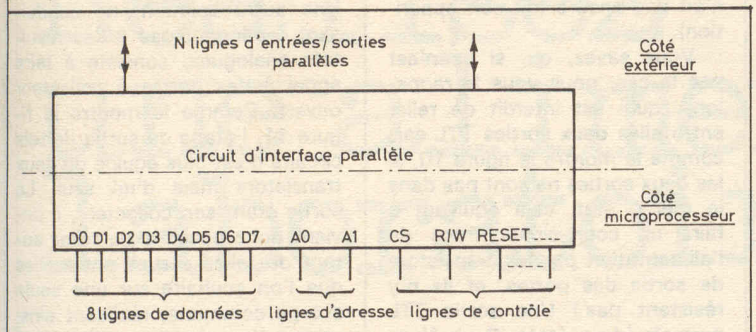


Fig. 7. – Synoptique d'un circuit d'interface parallèle.

les grandeurs analogiques à mesurer et les interfaces de sortie sont les circuits qui commandent les afficheurs que l'utilisateur du voltmètre a sous les yeux. Dans un ordinateur individuel classique, par contre, l'interface d'entrée est très souvent le circuit de couplage d'un clavier style « machine à écrire » et l'interface de sortie est, soit le circuit de couplage à une imprimante, soit le circuit de génération de signaux vidéo dans le cas d'un affichage sur récepteur TV.

Malgré cette diversité, les circuits d'interface se subdivisent en deux grandes familles principales qu'il est bon de connaître, au moins en général, ce sont les interfaces parallèles et les interfaces séries.

Les interfaces parallèles sont les plus simples à assimiler. Considérons à cet effet la figure 7 qui montre le synoptique très simplifié d'un tel circuit. Les données issues du microprocesseur entrent dans ce circuit ainsi qu'un certain nombre de lignes du bus de contrôle. Au moyen de ces lignes de contrôle, il est possible de ne faire rentrer dans le circuit que les données que l'on souhaite au moment où on le désire ; à partir de ce moment, et tant que l'on ne fait pas entrer de nouvelles données, celles-ci sont mémorisées dans le circuit et sont disponibles à sa sortie pour l'utilisateur. Ce procédé est en général réversible et les données que l'on applique côté « extérieur » du circuit peuvent être lues sur les lignes de données par le microprocesseur. Bien sûr, ces circuits sont en réalité très évolués et possèdent de nombreuses autres possibilités que nous décrirons en temps utile.

Le rôle d'un circuit d'interface série est un peu plus délicat à assimiler. Il faut savoir, au préalable, ce qu'est une liaison série et quel est son rôle. Imaginons que nous ayons un terminal, c'est-à-dire un ensemble constitué d'un clavier, d'une électronique et d'un écran de TV à relier à un ordinateur. Le terminal, tout comme l'ordinateur ne comprend que le binaire et travaille également et en général sur des mots de 8 bits. La première idée qui vient à l'esprit est de raccorder ces éléments au moyen d'une liaison parallèle ; il va donc nous falloir tirer 8 fils entre les deux plus une masse plus un ou deux signaux de contrôle, bref, cela va faire un joli câble. Si cette solution est applicable et peut parfois être appliquée, elle n'est que rarement choisie pour ce genre de connexion, en particulier lorsque la liaison est un tant soit peu longue. On lui préfère une liaison du type série, pour ce faire, le mot de 8 bits à envoyer rentre dans un circuit d'interface série, qui est en fait un convertisseur parallèle-série, et, comme le montre la figure 8, nos 8 bits se retrouvent à la queue leu leu sur un seul et unique fil. A l'arrivée, un circuit analogue effectue l'opération inverse et le tour est joué. Pour que cela fonctionne, il faut quand même prendre quelques précautions, en particulier, la vitesse de conversion série parallèle (c'est-à-dire la vitesse de transmission sur la ligne ou encore la fréquence de l'horloge qui entre dans le convertisseur) doit être la même à l'émission que celle qui effectue l'opération inverse à la réception, de plus pour différencier les mots de 8 bits qui se suivent les uns derrière les autres sur ce seul fil, il faut devant chaque mot de 8 bits un bit spécial appelé bit de start et derrière chaque mot de 8 bits, un bit spécial appelé bit de stop. Ces bits sont ajoutés automatiquement par le circuit d'interface série lorsqu'il travaille en émission et sont enlevés automatiquement par ce même circuit lorsqu'il travaille en réception. Par ailleurs, pour accroître la sécurité de la transmission, ces circuits peuvent faire seuls un certain nombre de contrôles dont ils peuvent rendre compte au microprocesseur. L'emploi de ces circuits étant assez répandu, nous aurons l'occasion de revenir plus longuement sur leur fonctionnement et sur leur principe. La figure 9 présente le synoptique simplifié d'un tel circuit. Remarquons simplement que du côté extérieur, en plus d'une ligne de sortie et d'une ligne d'entrée (ce circuit étant bidirectionnel), nous disposons de deux pattes pour définir l'horloge de transmission évoquée ci-avant ainsi que de deux pattes de signaux de contrôle qui, comme nous le verrons permettent, soit de connecter un MODEM, soit de travailler avec des terminaux lents de manière normale et sans que le microprocesseur n'ait à se soucier de quoi que ce soit.

Le bus de notre ordinateur individuel

Toutes ces considérations théoriques étant vues, nous allons commencer à nous rapprocher de la pratique avec la description des signaux du bus de notre système, bus que nous réaliserons, ainsi que l'alimentation, dès le mois prochain.

Notre bus est à un format compatible EXORciser, c'est-à-dire que toutes les cartes revêtues de ce label pourront être mises dans l'ordinateur sans difficulté ; cela n'est peut être pas très intéressant pour l'amateur mais pour ceux d'entre vous qui envisagent une utilisation professionnelle de la chose, sachez qu'il existe une centaine de

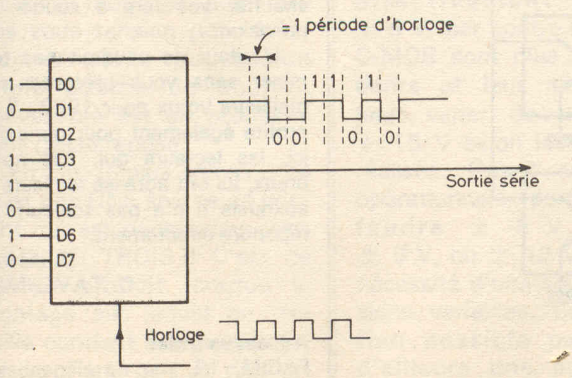


Fig. 8. – Principe général d'une liaison série : conversion parallèle-série.

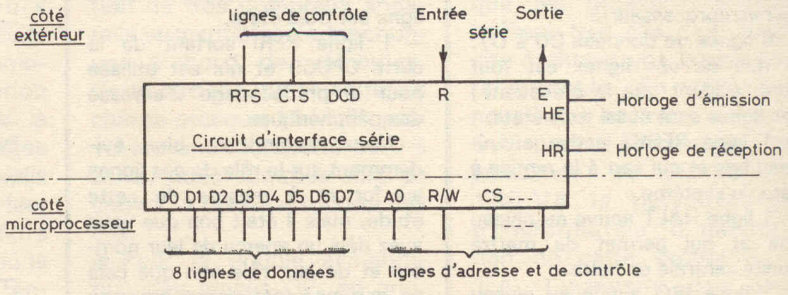


Fig. 9. – Synoptique d'un circuit d'interface série.

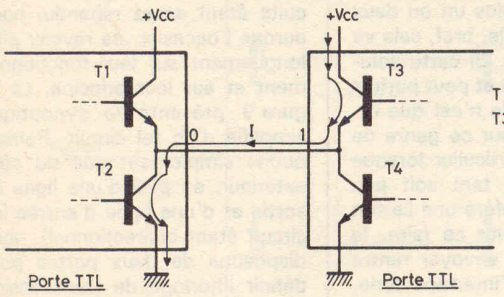


Fig. 10. – Voici pourquoi il ne faut pas relier deux sorties de portes TTL classiques entre elles.

cartes à ce format, tant chez Motorola avec sa gamme Micromodules que chez EFCIS ou d'autres fabricants français et étrangers. Par ailleurs, et pour cette raison, les cartes de l'ancien mini-ordinateur sont strictement compatibles des cartes du nouveau système. Le format des cartes proprement dit a par contre été réduit par rapport aux cartes EXORciser ; cette décision avait été prise pour l'ancien mini-ordinateur et nous l'avons conservée pour maintenir la compatibilité, même sur le plan mécanique.

Le bus est équipé de connecteurs encartables de deux fois 43 contacts au pas de 3,96 mm comme nous le verrons le mois prochain, les 86 lignes ainsi définies ne sont cependant pas toutes utilisées ni même définies. Nous allons voir ci-après celles que nous allons utiliser :

– 18 lignes d'adresses A0 à A18, le rôle de ces lignes se passe de commentaire ; ces lignes peuvent passer en trois états (voir plus avant) ; ces lignes sortent de la carte CPU09 (donc du microprocesseur).

– 8 lignes de données D0 à D7, le rôle de ces lignes est tout aussi évident que le précédent ; ces lignes sont aussi trois états.

– 1 ligne RESET active au niveau bas et qui sert à la remise à zéro du système.

– 1 ligne HALT active au niveau bas et qui permet de mettre l'unité centrale en halte.

– 1 ligne IRQ active au niveau bas qui est une entrée d'interruption masquable sur l'unité centrale (notion définie ultérieurement).

– 1 ligne FIRQ active au niveau bas qui est une ligne d'interruption rapide masquable sur l'unité centrale (idem).

– 1 ligne NMI active au niveau bas qui est une ligne d'interruption non masquable sur l'unité centrale (idem).

– 1 ligne E ou phi2 qui est une sortie d'horloge de l'unité centrale, cette ligne sert de référence de temps à tous les circuits du système comme nous le verrons par la suite ; cette ligne n'est pas trois états.

– 1 ligne VMA qui est une sortie de l'unité centrale indiquant la validité ou non des adresses présentes sur les lignes A0 à A18 ; cette ligne est active au niveau haut et n'est pas trois états.

– 2 lignes BA et BS qui sont des sorties de l'unité centrale indiquant dans quel état se trouve celle-ci (normal, en halte, en reconnaissance d'interruption, etc.) ; ces lignes ne sont pas trois états.

– 1 ligne R/W ou lecture écriture qui est une sortie de l'unité centrale et qui indique si celle-ci lit (R/W = 1) ou écrit (R/W = 0) dans la mémoire ; cette ligne est trois états.

– 1 ligne PERI sortant de la carte CPU09 et qui est utilisée pour le prédécodage d'adresse des périphériques.

Nous reviendrons bien évidemment sur le rôle de ces lignes au fur et à mesure de cette étude, mais il était bon que vous ayez déjà un aperçu de leur nombre et de leur rôle afin que cela ne soit pas totalement nouveau lorsque nous vous présenterons le schéma de la carte unité centrale. Nous allons cependant préciser ce que l'on entend par ligne trois états et ligne à collecteur ouvert (bien que cette dernière n'ait pas encore fait son apparition).

Vous savez, ou, si ce n'est pas le cas, nous vous le rappelons, qu'il est interdit de relier entre elles deux sorties TTL car, comme le montre la figure 10, si les deux sorties ne sont pas dans le même état, cela équivaut à faire un court-circuit franc de l'alimentation par les transistors de sortie des portes, et ils n'y résistent pas ! Une sortie TTL normale (deux états, 0 ou 1) se prête donc très mal à la notion de bus évoquée en début d'article puisque nous avons vu que toutes les lignes de même nom étaient reliées entre elles. Il est donc fait appel, dans tout circuit servant en micro-informatique à des sorties « trois états » (ce qui est un comble en binaire !). Les circuits équipés de telles sorties présente la propriété de voir celles-ci passer en haute impédance (ce qui équivaut à une déconnexion) sous l'action d'un signal appliqué à une patte adéquate du circuit, patte ayant généralement pour nom CE (Chip Enable), CS (Chip Select), OE (Output Enable) ou encore tout simplement E (Enable), « enable » signifiant autoriser ou valider. Sous réserve d'une gestion logique de ces pattes d'activation, il devient possible de connecter autant de sorties trois états entre elles qu'on le désire.

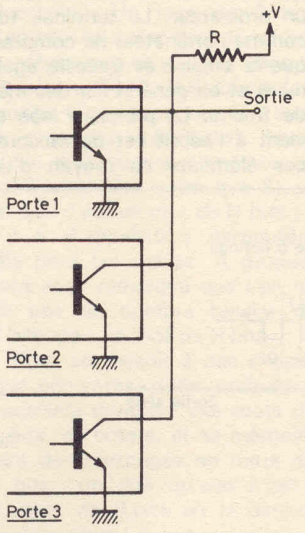


Fig. 11. – Connexion entre elles de plusieurs sorties de portes à collecteur ouvert.

Une autre solution, ne conduisant cependant pas à des résultats analogues, consiste à faire appel à des portes à collecteur ouvert. Comme le montre la figure 11, l'étage de sortie de tels circuits n'est plus équipé de deux transistors mais d'un seul. La sortie étant son collecteur, il devient donc possible de relier autant de telles sorties entre elles que l'on souhaite sur une seule charge commune, réalisant ainsi ce que l'on appelle un OU câblé (il suffit en effet qu'une quelconque des sorties soit à zéro pour que la sortie globale soit à zéro, mais nous y reviendrons...).

Conclusion

Nous allons nous arrêter là pour aujourd'hui, les notions théoriques que nous avons évoquées étant assez importantes et assez nombreuses pour nous permettre d'aborder sans difficulté la première phase de cette réalisation.

Nous vous conseillons de ne pas chercher à trop approfondir ce que nous avons exposé, nous y reviendrons en effet lorsque le moment sera opportun ; de même, ne vous inquiétez pas si quelques notions vous ont échappé, cela ne vous empêchera pas de mener à bien la réalisation de votre ordinateur individuel, et, lorsque vous l'aurez entre les mains vous verrez que la micro-informatique, mais c'est très simple... ou presque.

Le mois prochain, nous parlerons boîtier, nous réaliserons le circuit imprimé de fond de panier, c'est-à-dire celui qui supporte ce fameux bus que nous venons de présenter, et nous réaliserons aussi l'alimentation ; faites chauffer vos fers à souder en prévision...

L'auteur ne voudrait pas terminer sans vous présenter ses meilleurs vœux pour 1982 ; il en profite également pour remercier ici, les lecteurs qui, fort nombreux, lui ont adressé les leurs et auxquels il n'a pas toujours pu répondre directement.

(A suivre.)

C. TAVERNIER

Adresses utiles :
FACIM, 19, rue de Hegenheim, 68300 Saint-Louis.
INCODEC, 9, chemin de Laprat, 26000 Valence.